ГЛАВА 3

Редактор анимации

В этой главе будут представлены инструменты анимации, доступные в Expression Blend. Если у вас имеется некоторый опыт создания вручную анимационных последовательностей в формате XAML при разработке приложений в Visual Studio, то вам должно быть известно, насколько трудоемким может оказаться этот процесс[1]. К счастью, в Expression Blend интегрирован довольно развитый редактор анимации, в котором можно без особого труда фиксировать изменения состояния объектов, используя простой принцип временной шкалы. Читая эту главу, вы узнаете, каким образом анимация образуется из *раскадровок* и *ключевых кадров*. Каждый ключевой кадр анимации отвечает за изменение значения указанного свойства целевого объекта.

Научившись сначала пользоваться основными инструментами анимации, вы затем откроете для себя назначение эффектов инерционности движения в анимации. По существу, *эффекты инерционности движения* упрощают внедрение физических явлений в анимацию, в том числе отскакивание, притягивание и упругость. Кроме того, в состав Expression Blend входит редактор ключевых сплайнов, в котором можно точнее изменять интерполируемые значения (т.е. эффекты ускорения) при подходе к заданному ключевому кадру или выходе из него.

И в заключение главы будет вкратце изложен вопрос, более подробно рассматриваемый в главе 4, а именно: назначение разных *объектов поведения*. Здесь вы ознакомитесь с объектом поведения типа `ControlStoryboardAction`, благодаря которому взаимодействие с раскадровкой полностью реализуется в разметке.

Назначение служб анимации

Как отмечалось в главе 2, ясное представление о том, как следует работать с графикой, имеет решающее значение для успешной разработки приложений на платформах WPF и Silverlight, независимо от того, что по этому поводу думают сами разработчики. Аналогично, следует отметить важную роль служб анимации при построении реальных приложений WPF и Silverlight промышленного уровня. Что бы вы ни думали по этому поводу, применение служб анимации никак не связано с процессом создания видеоигр и приложений, насыщенных мультимедийным содержимым, хотя службы анимации будут, очевидно, полезными и в этих случаях.

На платформе WPF или Silverlight под *анимацией* подразумевается простое изменение значения свойства объекта в течение времени. Так, если требуется управлять

[1] Если вам приходилось когда-нибудь создавать анимацию в других средах разработки, вы будете приятно удивлены, насколько просто это делается на платформах WPF и Silverlight. В подавляющем большинстве случаев теперь уже нет необходимости создавать потоки вручную, стирать и перерисовывать изображения, создавать закадровые буферы и выполнять неприятные расчеты прямоугольников.

изменением цвета фона объекта от ярко-зеленого до темно-зеленого оттенка в течение пяти секунд, можно воспользоваться *анимацией кисти*. А если требуется перемещать специальный графический элемент по неподвижной геометрической линии, то в этом случае можно воспользоваться *анимацией по траектории*. Читая эту и остальные главы данной книги, вы обнаружите, что практически любое свойство, доступное на панели Properties в среде Expression Blend, может стать целевым для служб анимации.

Область применения служб анимации

Подобно графике, анимация появляется в самых неожиданных местах. Как будет показано в главе 5, чаще всего анимация применяется для внедрения визуальных подсказок в специальные стили оформления или шаблоны элементов управления. Например, с помощью анимации можно легко определить внешний вид специального элемента управления, когда курсор наводится на него, когда курсор покидает его или же когда производится щелчок кнопкой мыши на его поверхности. Помимо этого, можно определить дополнительные виды анимации, определяющие внешний вид элемента управления, когда он получает логический фокус, теряет фокус и в остальных случаях.

Анимацию можно использовать для четкого и плавного перехода между значениями свойств объектов. В качестве примера можете создать объект типа `Window` в проекте приложения на платформе WPF и осуществить анимацию вращения диспетчера компоновки (со всеми элементами управления, которые он содержит) в трехмерной плоскости. А в проект приложения на платформе Silverlight можете добавить анимацию переворачивания страницы как при просмотре книжки с иллюстрациями. Если же вы занимаетесь созданием видеоигры, воспользуйтесь анимацией для перемещения стен лабиринта, вывода вспышкой на экран сообщения “Игра окончена” и т.д. Следует иметь в виду, что применение служб анимации стало обычным явлением в проектах приложений на платформах WPF и Silverlight, и в среде Expression Blend IDE подобные визуальные эффекты достигаются довольно просто.

Рабочее пространство анимации в Expression Blend

Как пояснялось в главе 1, панель Objects and Timeline служит для редактирования объекта, выбранного на монтажном столе. Оказывается, что та же самая панель служит местом доступа к редактору анимации в Expression Blend. Для того чтобы приступить к изучению средств создания анимации в среде Expression Blend IDE, создайте новый проект приложения Silverlight, присвоив ему имя `SimpleBlendAnimations`[2].

Первым шагом на пути к созданию анимации в Expression Blend должен стать выбор одного или нескольких объектов в качестве целевых для логики анимации. В настоящий момент во вновь созданном проекте приложения Silverlight имеется только объект `LayoutRoot` типа `Grid`, а также объект типа `UserControl`, определяющий собственно пользовательский интерфейс. Конечно, можно осуществить анимацию свойств и этих объектов, но рассматриваемый здесь пример окажется более интересным и полезным, если добавить новый объект на монтажном столе. С этой целью можете ввести объект любого типа, будь то `Button`, `Rectangle` или контур, нарисованный инструментом Pen либо Pencil, но ради примера создайте простой объект типа `Ellipse`, присвоив ему имя

[2] В прикладном интерфейсе WPF API поддерживается ряд дополнительных средств создания анимации, недоступных на платформе Silverlight. Способы, рассматриваемые в первом примере проекта, одинаково применимы для создания анимации независимо от используемого прикладного интерфейса API конкретной платформы.

myCircle (мой круг) и подобрав по своему вкусу сплошной цвет заливки для его свойства Fill (подробнее о том, как это делается в редакторе кистей, см. в главе 2).

Когда вы пользуетесь инструментами анимации в Expression Blend, то, скорее всего, стремитесь изменить компоновку рабочего пространства анимации, активизируемого с помощью команды меню Window⇨Workspaces (Окно⇨Рабочие пространства) или функциональной клавиши <F6>. Нажмите эту клавишу, чтобы увидеть, каким образом реорганизуются панели в рабочем окне среды Expression Blend IDE, а панель Objects and Timeline располагается вдоль нижнего края этого окна (рис. 3.1).

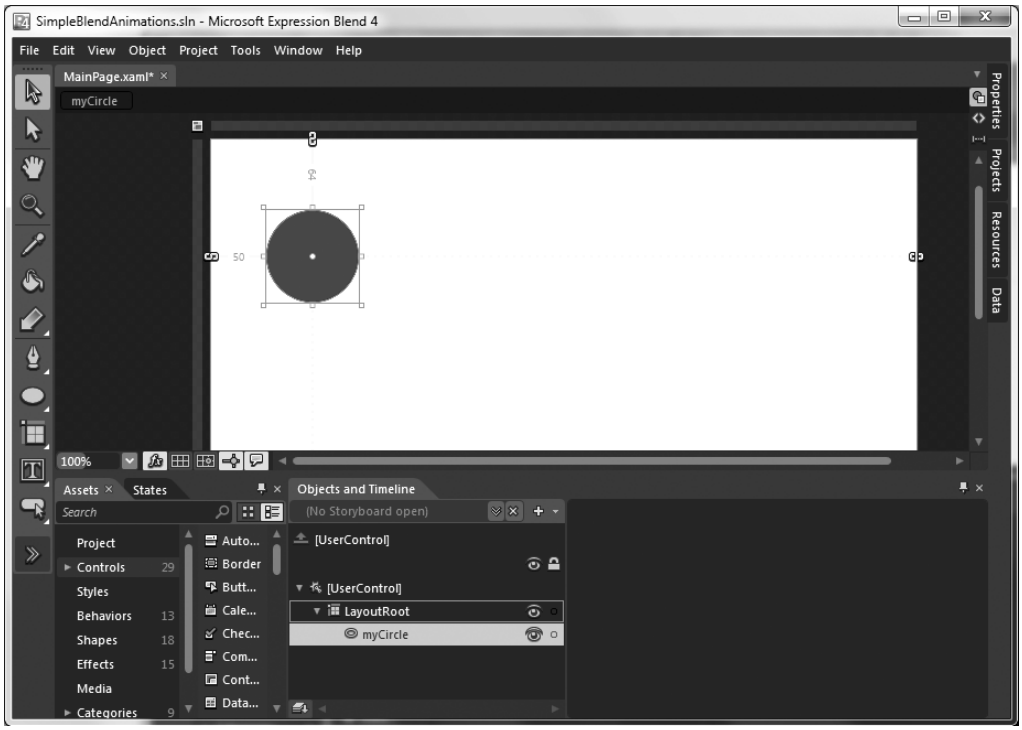

Рис. 3.1. Компоновка рабочего пространства анимации

Примечание. Нажимая функциональную клавишу <F6>, можете переходить от стандартного рабочего пространства конструирования (Design) к рабочему пространству анимации (Animation) и обратно.

Создание новой раскадровки

Анимация на платформе WPF или Silverlight фиксируется в формате XAML с использованием *раскадровки*. Следовательно, когда возникает потребность в получении новой анимационной последовательности, этот процесс вы должны начинать с создания новой раскадровки на панели Objects and Timeline. С этой целью щелкните на кнопке New со знаком + на пиктограмме, как показано на рис. 3.2. (В данный момент не имеет особого значения, какой именно объект выбран в иерархическом представлении объектов на панели Objects and Timeline.)

Как только вы щелкнете на этой кнопке, вам будет предложено присвоить новой раскадровке особое имя, например AnimateCircle (анимация круга), (рис. 3.3).

Присваивание имени раскадровке имеет большое значение, поскольку по этому имени вам придется манипулировать анимацией как в коде, так и в разметке XAML.

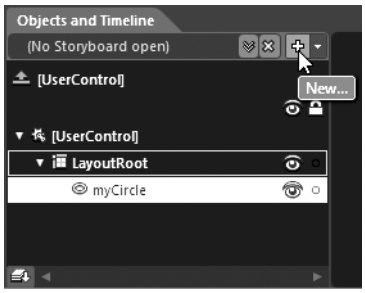

Рис. 3.2. Создание новой раскадровки

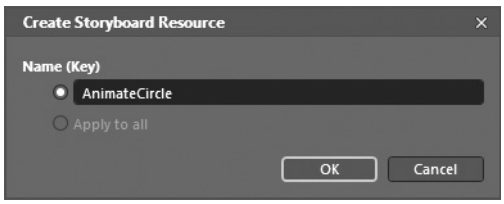

Рис. 3.3. Именование раскадровки

Следует также иметь в виду, что новая раскадровка анимации сохраняется в Expression Blend в качестве ресурса (см. главу 2) в текущем объекте типа `UserControl` на платформе Silverlight или объекте типа `Window` на платформе WPF. Так, если вы откроете окно для просмотра содержимого монтажного стола в коде XAML, то обнаружите разметку, выделенную на рис. 3.4.

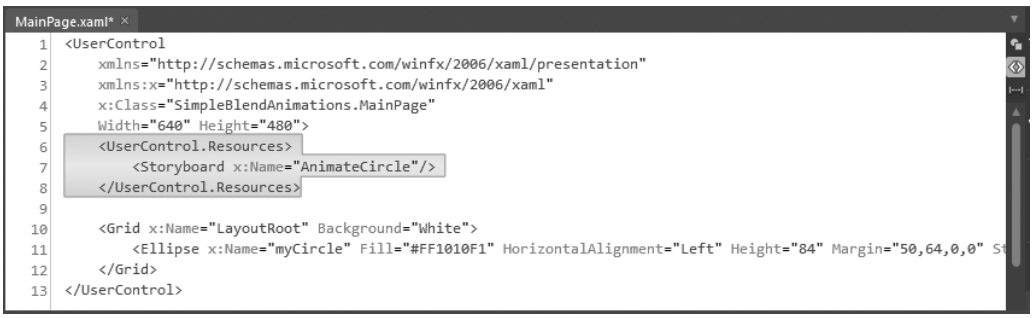

Рис. 3.4. Раскадровки сохраняются в виде ресурсов, формируемых на уровне документов

Примечание. В отличие от типичных ресурсов объектов, рассматривавшихся в главе 2, раскадровки тесно связаны с объектом, для анимации которого они предназначены. Следовательно, создавать раскадровку на уровне всего приложения нецелесообразно. Если вы предполагаете пользоваться созданной анимационной последовательностью неоднократно, то разместите соответствующую раскадровку в специальном стиле оформления, шаблоне или объекте типа `UserControl`, чтобы впоследствии извлечь из него содержащуюся в нем анимацию. Более подробно эти вопросы рассматриваются в главе 5.

Управление имеющимися раскадровками

Создав новый объект раскадровки, например `AnimateCircle`, вы сразу же обнаружите его в списке, раскрывающемся на панели Objects and Timeline. В связи с тем что в одном объекте типа `Window` или `UserControl` можно нередко обнаружить довольно большое количество раскадровок, отвечающих за анимацию разных объектов, отдельную раскадровку следует выбирать для правки, используя именно этот раскрывающийся список на панели Objects and Timeline.

Следует также иметь в виду, что раскадровки можно искать по имени аналогично поиску нужных средств в библиотеке ресурсов и на панели Properties, выбрав вариант Search из того же самого раскрывающегося списка, как показано на рис. 3.5.

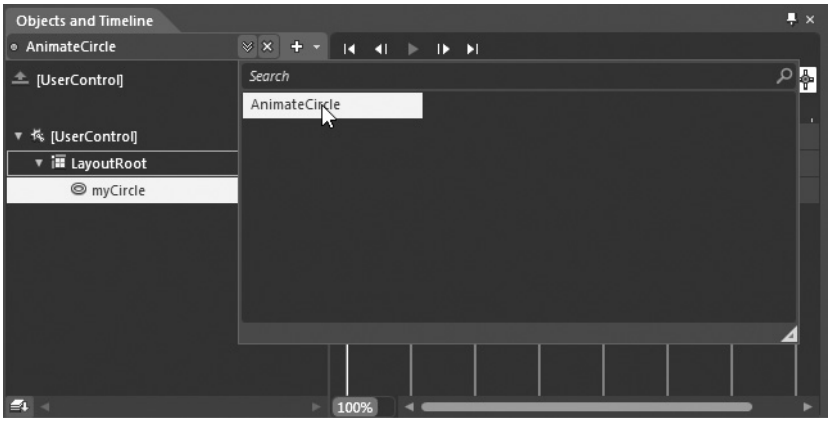

Рис. 3.5. Отдельные раскадровки можно выбрать из раскрывающегося списка или найти в режиме поиска

И еще одно, последнее вводное замечание: раскадровку можно легко переименовать, скопировать или удалить, щелкнув на ней правой кнопкой мыши и выбрав соответствующую команду из всплывающего контекстного меню (рис. 3.6). Так, если вы выберете из этого меню команду Reverse, последовательность ключевых кадров анимации изменится на обратную, что, согласитесь, очень удобно!

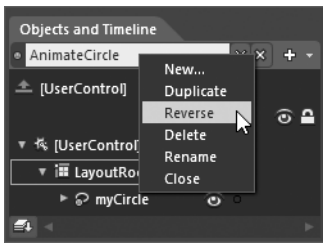

Рис. 3.6. Имеющимися раскадровками можно управлять на панели Objects and Timeline

Ввод ключевых кадров анимации

Создав новую раскадровку или выбрав уже имеющуюся для последующей правки, можете приступать к вводу любого количества ключевых кадров анимации в редакторе временной шкалы. Исследуя текущий редактор временной шкалы, обратите внимание на желтую вертикальную линию напротив нульсекундной отметки анимации. Эта линия обозначает текущий момент времени анимации в данной раскадровке. Прямо над этой желтой линией находится яйцевидная пиктограмма со знаком + справа от нее. Она обозначает кнопку Record Keyframe (Записывать ключевой кадр). Если щелкнуть на этой кнопке, в текущий момент времени анимации будет введен ключевой кадр.

Вводя ключевой кадр, вы должны держать в уме объект, выбранный в настоящий момент на панели Objects and Timeline. Дело в том, что в одной раскадровке можно управлять анимацией целого ряда объектов, находящихся на монтажном столе, а

следовательно, в один и тот же момент времени может присутствовать несколько ключевых кадров анимации. Что же касается рассматриваемого здесь примера, то непременно выберите созданный вами ранее объект типа `Ellipse` и щелкните на кнопке Record Keyframe, чтобы ввести новый ключевой кадр на нульсекундной отметке анимации, как показано на рис. 3.7.

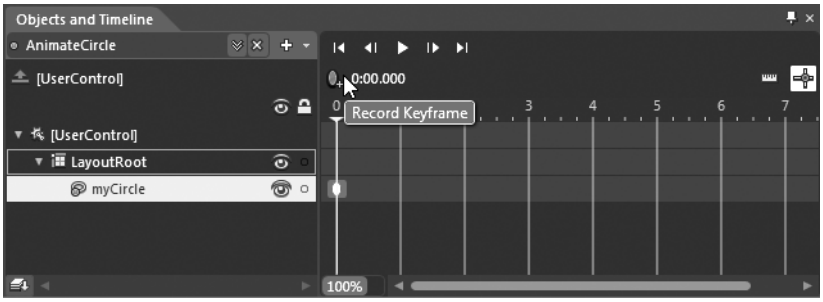

Рис. 3.7. Ввод ключевых кадров на временной шкале

Введя ключевой кадр анимации, обратите внимание то, что теперь монтажный стол обрамлен красной рамкой, а в левом верхнем углу монтажного стола появился небольшой элемент управления, с помощью которого можно включать и выключать режим записи анимационной последовательности (рис. 3.8).

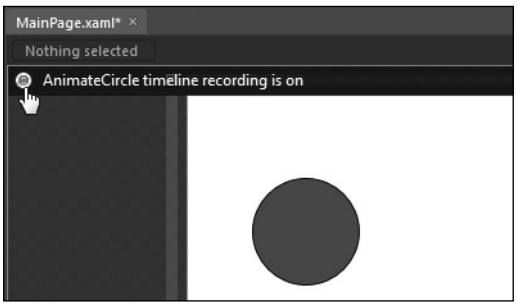

Рис. 3.8. На монтажном столе можно включать и выключать режим записи анимационной последовательности

Примечание. Отдельные ключевые кадры могут быть перемещены в редакторе временной шкалы стандартным методом перетаскивания мышью. Так, если вы ввели ключевой кадр на трехсекундной отметке анимации, но хотите установить его на шестисекундной отметке, можете легко сделать это, перетащив ключевой кадр к нужной отметке анимации. Кроме того, можете удалить или скопировать ключевой кадр, щелкнув на нем правой кнопкой мыши и выбрав соответствующую команду из всплывающего контекстного меню.

Фиксация изменений в свойствах объектов

После ввода первого ключевого кадра, который совсем *не* обязательно устанавливать в нулевой, но в любой удобный для анимации момент времени в зависимости от ее характера, вам нужно ввести хотя бы еще один, дополнительный ключевой кадр, чтобы зафиксировать вместе с первым ключевым кадром промежуток времени, в котором будет происходить данная анимация. Для перемещения по временной шкале щелкните на небольшой

треугольной “шапке” над желтой вертикальной линией и перетащите ее мышью или же щелкните непосредственно на нужном месте временной шкалы. Используя любой из этих двух способов, введите второй ключевой кадр на двухсекундной отметке анимации, т.е. щелкните еще раз на кнопке с яйцевидной пиктограммой, как показано на рис. 3.9.

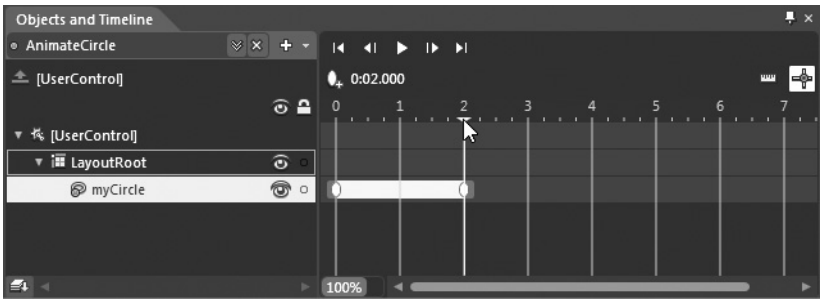

Рис. 3.9. Определение двухсекундного промежутка времени анимации

Далее следует самое интересное. Убедитесь сначала в том, что на монтажном столе выбран объект типа `Ellipse`, а затем воспользуйтесь панелью Properties и монтажным столом, чтобы внести ряд изменений в состояние этого объекта. В частности, можете выделить объект типа `Ellipse` и переместить его в новое положение на монтажном столе. При этом появится небольшая пунктирная линия, обозначающая расстояние, на которое был перемещен объект. Кроме того, можете выбрать новый цвет заливки в свойстве `Fill` объекта типа `Ellipse` посредством редактора кистей или внести изменения в свойства `Height` и `Width`, перейдя к области Layout на панели Properties. Аналогичным образом внесите еще три или четыре изменения в состояние объекта типа `Ellipse`. На рис. 3.10 показано, как выглядит объект `myCircle` по достижении двухсекундной отметки анимации.

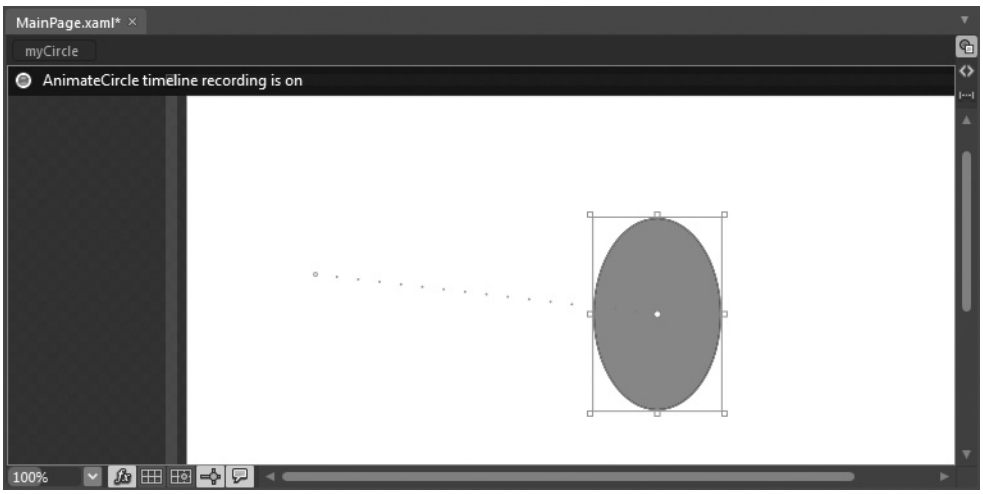

Рис. 3.10. Возможный вид объекта `myCircle` по достижении двухсекундной отметки анимации (у вас он может выглядеть иначе)

Проверка анимации

Теперь можете проверить созданную вами анимацию, щелкнув на кнопке Play (Воспроизвести), находящейся в области инструментов на временной шкале (рис. 3.11).

В итоге окружность плавно перейдет из своего исходного состояния в конечное на протяжении двух секунд анимации.

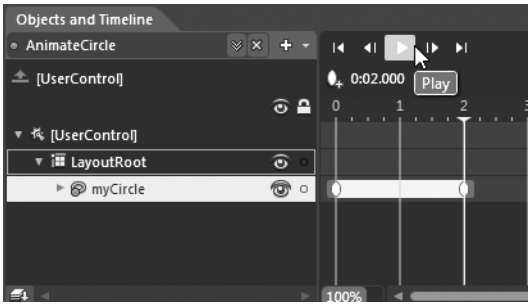

Рис. 3.11. Проверка созданной анимации в режиме воспроизведения

Упомянутые выше инструменты действуют аналогично органам управления на типичном цифровом универсальном проигрывателе и позволяют переходить от предыдущих кадров анимации к следующим, а также к первому и последнему кадру раскадровки. Уделите время опробованию этих инструментов и научитесь перемещать желтую вертикальную линию по временной шкале, захватив и перетащив мышью ее "шапку" или просто щелкнув непосредственно в нужном месте временной шкалы.

Просмотр разметки анимации

Прежде чем двигаться дальше, уделите время просмотру исходной разметки, автоматически сформированной в коде XAML средствами Expression Blend IDE. Как видите, элемент разметки `<Storyboard>` заполнен целым рядом новых подчиненных элементов, определяющих порядок анимации объекта типа `Ellipse`. Не вдаваясь в подробности этой разметки, следует все же обратить внимание на то, что объекту анимации (в данном случае `ColorAnimationUsingKeyFrames`) известно, свойства каких именно объектов должны изменяться и в какие именно моменты анимации это должно происходить. Для первой цели служат свойства `TargetProperty` (Свойство целевого объекта) и `TargetName` (Имя целевого объекта), а для второй — свойство `KeyTime` (Время наступления ключевого кадра) объекта анимации.

```
<ColorAnimationUsingKeyFrames
  Storyboard.TargetProperty="(Shape.Fill).(SolidColorBrush.Color)"
  Storyboard.TargetName="myCircle">
  <EasingColorKeyFrame KeyTime="0" Value="#FF1010F1"/>
  <EasingColorKeyFrame KeyTime="0:0:2" Value="#FFF110DE"/>
</ColorAnimationUsingKeyFrames>
```

Настройка свойств раскадровки

Если выбрать объект раскадровки из раскрывающегося списка на панели Objects and Timeline (см. рис. 3.5), то свойства самой раскадровки можно настроить на панели Properties[3]. Как показано на рис. 3.12, раскадровка может быть настроена на режимы автоматического изменения на обратное направления воспроизведения анимации. Так, если установить флажок свойства `AutoReverse` (Автоматически изменить направление

[3] Имеется также возможность настраивать отдельные ключевые кадры. О том, как это сделать, речь пойдет ниже, когда будет рассматриваться назначение эффектов инерционности движения в анимации.

на обратное), вся анимация займет уже четыре секунды: по две секунды в каждом направлении ее воспроизведения.

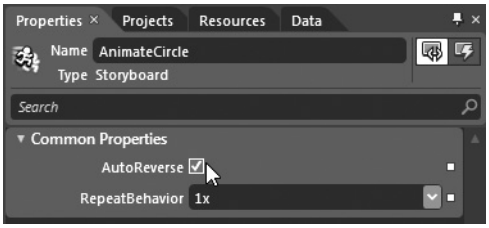

Рис. 3.12. Активизация режимов автоматического изменения на обратное направление воспроизведения анимации

Имеется также возможность настроить режим повторения анимации, выбрав из раскрывающегося списка свойства `RepeatBehavior` один следующих четырех вариантов: `1x` (Однократно), `2x` (Двукратно), `3x` (Трехкратно) и **`Forever`** (Бесконечно). Но вы вольны ввести любое число повторений анимации, дополнив его буквой **`x`**. Так, если вам требуется воспроизвести анимацию пять раз, введите **`5x`** в текстовом поле свойства `RepeatBehavior`.

В текстовом поле свойства `RepeatBehavior` можно также ввести числовое значение, обозначающее конкретное *время* воспроизведения анимации, а не число ее повторений. Для того чтобы указать единицу изменения времени, введите заданное время в приведенном ниже формате.

```
дни:часы:минуты:секунды:доли_секунды
```

Указывать `дни` и `доли_секунды` необязательно. Как правило, заданное время указывается в следующем обобщенном формате:

```
часы:минуты:секунды
```

Так, если требуется, чтобы анимация воспроизводилась в течение 15 секунд, а затем прекращалась полностью независимо от того, осуществляется ли в ней циклический проход всех ключевых кадров, следует ввести значение **`00:00:15`** в текстовом поле свойства `RepeatBehavior`, как показано на рис. 3.13.

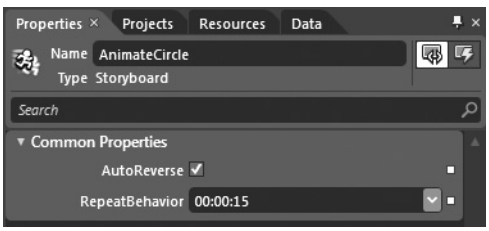

Рис. 3.13. Указание времени воспроизведения анимации, включая ее повторение, если таковое имеет место

Изменение масштаба временной шкалы

Прежде чем переходить к рассмотрению способов взаимодействия с раскадровками в коде, следует отметить еще одну особенность редактора анимации в Expression Blend. В левом нижнем углу этого редактора находится элемент, управляющий измене-

нием масштаба временной шкалы. По умолчанию в нем установлен масштаб 100%, а следовательно, на временной шкале продолжительность и отдельные моменты времени анимации указываются в секундах. Если же требуется более мелкая градация времени для установки ключевых кадров в моменты, обозначающие доли секунды, или, наоборот, более крупная градация времени, измените масштаб временной шкалы по своему усмотрению. В качестве примера на рис. 3.14 показано изменение масштаба временной шкалы до 750%, чтобы разделить анимацию на доли секунды.

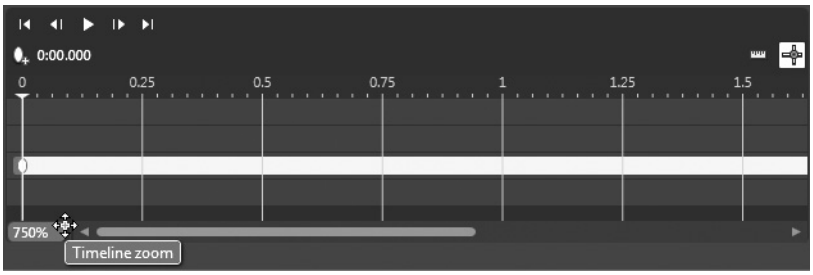

Рис. 3.14. Увеличение масштаба временной шкалы до более мелких градаций времени анимации

Примечание. Если вы изменили масштаб временной шкалы, вновь сделайте его равным 100%, чтобы вернуться к исходному виду этой шкалы с градацией по секундам.

Взаимодействие с раскадровками в коде

Когда вы создаете анимацию в Expression Blend и делаете это в проекте приложения Silverlight, то при запуске этого приложения на выполнение никакой логики, начинающей анимацию по раскадровке, автоматически не добавляется. С другой стороны, при запуске на выполнение приложения WPF анимация автоматически начинается по раскадровке, когда главное окно приложения загружается в оперативную память, хотя это положение можно впоследствии изменить. О том, как изменяется этот устанавливаемый по умолчанию режим работы приложения WPF, речь пойдет далее, а до тех пор покажем на рассматриваемом здесь примере проекта, как можно начинать анимацию посредством кода. Что же касается управления раскадровками в разметке, то мы вернемся к этому вопросу, когда будем рассматривать объект поведения типа `ControlStoryboardAction`.

Прежде всего необходимо решить, какое именно действие пользователя должно привести к началу анимации. Этим действием может стать щелчок на кнопке, выбор пункта меню, щелчок на самом круге, нажатие клавиши или любой другой, специально указываемый вид входной информации от пользователя. В целях рассматриваемого здесь примера допустим, что анимация должна начинаться в том случае, если пользователь щелкнет на элементе управления типа `Button`. Итак, найдите объект типа `Button` на панели Tools, но не забывайте, что искать нужные вам элементы управления можно также в библиотеке ресурсов и на панели Assets.

Примечание. Текст надписи на выбранном объекте типа `Button` можно изменить, соответственно настроив его свойство `Content` (Содержимое) в области Common Properties на панели Properties.

Добавив объект типа `Button` на монтажный стол, переименуйте новый элемент управления пользовательского интерфейса на `btnStartAnimation` (кнопка начала анима-

ции), воспользовавшись панелью Properties. Далее щелкните на кнопке Events, находящейся на этой же панели, и найдите событие Click, связанное с выбранным объектом типа Button[4]. Дважды щелкните на текстовом поле справа от имени события Click. Теперь в исходном коде текущего проекта вы обнаружите приведенный ниже пустой обработчик событий.

```
private void btnStartAnimation_Click(object sender,
  System.Windows.RoutedEventArgs e)
{
  // Что сделать: добавить здесь реализацию обработчика событий в коде.
}
```

Напомним, что в среде Expression Blend IDE элементы раскадровки автоматически сохраняются в качестве ресурса объекта, размещаемого в словаре ресурсов соответствующего объекта типа Window на платформе WPF или же объекта типа UserControl на платформе Silverlight. С учетом этого обстоятельства ваша первая задача по взаимодействию с раскадровками в коде состоит в том, чтобы найти объект раскадровки в коллекции ресурсов по ключевому имени, присвоенному раскадровке при ее создании (это имя AnimateCircle, если вы следовали рассматриваемому здесь примеру). Найдя этот объект, вызовите метод Begin(). Ниже приведен полный код обработчика событий Click[5].

```
private void btnStartAnimation_Click(object sender,
  System.Windows.RoutedEventArgs e)
{
  Storyboard animCircle;
  animCircle = (Storyboard)this.Resources["AnimateCircle"];
  animCircle.Begin();
}
```

Запустите свое приложение на выполнение, щелкните на элементе управления типа Button и посмотрите, как воспроизводится анимация!

Подробнее о классе Storyboard

Вы, вероятно, уже догадались, что в классе Storyboard определено намного больше функциональных возможностей, чем те, которыми обладает метод Begin(). В частности, в этом классе предоставляются методы Pause() и Stop() для частичной и полной остановки анимации соответственно, а также целый ряд свойств, с помощью которых определяются те же самые режимы повторения и автоматического изменения на обратное направление воспроизведения анимации, что и рассматривавшиеся ранее на панели Properties. Если вас интересует полное описание класса Storyboard, обращайтесь за справкой к документации на .NET Framework 4.0 SDK или Silverlight SDK.

Примечание. В исходный код рассматриваемого здесь примера проекта включено создание ряда других объектов кнопок для дополнительного управления анимацией по раскадровке. Надеюсь, этот код не окажется для вас слишком сложным.

На этом краткое введение в способы и средства создания анимации в среде Expression Blend IDE завершается. Нам еще предстоит рассмотреть целый ряд интересных вопро-

---

[4] В главе 2 демонстрировался процесс обработки на панели Properties событий, наступающих для заданного объекта. (Помните о кнопке с изображением молнии на пиктограмме?)

[5] В том файле исходного кода, где используется класс Storyboard, необходимо импортировать пространство имен System.Windows.Media.Animation, что, как правило, делается автоматически в среде Expression Blend IDE.

сов, поэтому если вы вполне освоились с представленными выше основными инструментами для создания анимации в Expression Blend, смело переходите к чтению следующей части этой главы.

Исходный код. Исходный код примера проекта `SimpleBlendAnimations` находится в папке `Ch 3 Code` загружаемого архива примеров проектов к данной книге.

Способы анимации, характерные для платформы WPF

В прикладном интерфейсе Windows Presentation Foundation API поддерживается ряд полезных способов анимации, которые, к сожалению, не находят поддержки в текущей версии платформы Silverlight. В первом из этих способов применяются *траектории движения*, с помощью которых можно легко перемещать элементы пользовательского интерфейса по объекту типа `Path`, нарисованному инструментом Pen или Pencil. И хотя аналогичного результата можно, конечно, добиться и в приложении Silverlight, для этого потребуется значительно больше усилий.

Второй характерный для платформы WPF способ анимации состоит в применении *триггеров* для взаимодействия с раскадровкой. На платформе WPF триггер обозначает применявшийся некогда способ реагирования на условие наступление события в разметке XAML. Например, с помощью триггеров можно написать такую разметку, в которой отслеживаются различные события от мыши (наведение курсора мыши на объект, нажатие и отпускание кнопки мыши и т.д.), события логического фокуса, события от клавиатуры и пр. Кроме того, можно написать дополнительную разметку, в которой отдельная раскадровка будет начинаться при наступлении подобных событий. Вам, возможно, известно, что среда триггеров на платформе WPF первоначально предназначалась для того, чтобы разработчики могли внедрять визуальные подсказки в специальные шаблоны с целью переопределить пользовательский интерфейс в отношении элементов управления и форм.

Несмотря на весьма ограниченную поддержку триггеров на платформе Silverlight, по тому же самому принципу действует диспетчер визуальных состояний (VSM). В действительности диспетчер VSM был настолько хорошо принят в программистских кругах, что на платформе WPF был внедрен свой вариант VSM в ее прикладной интерфейс API, начиная с версии .NET 4.0. С учетом этого обстоятельства у разработчиков приложений на платформе WPF теперь имеются два варианта выбора, когда дело доходит до внедрения визуальных подсказок в специальные шаблоны: триггеры или диспетчер VSM. Подробнее о том, как пользоваться триггерами и диспетчером VSM для построения специальных шаблонов, речь пойдет в главе 5, а до тех пор уделим основное внимание только триггерам для управления независимыми объектами раскадровки. Но сначала рассмотрим назначение траекторий движения, применяемых для анимации на платформе WPF.

Примечание. Как будет показано в конце главы, объект поведения типа `ControlStoryboardAction` дает также возможность манипулировать анимацией в разметке, используя средства WPF или Silverlight.

Работа с траекториями движения на платформе WPF

В Expression Blend предоставляется очень удобная возможность автоматически формировать на платформе WPF раскадровку, в которой выбранный объект должен перемещаться по предварительно заданной траектории движения. В частности, траектории

движения можно использовать для перемещения специальных графических элементов по вершинам линейного графика, показывая прибыли компании, что, конечно, приятнее, чем убытки, а возможно, и для перемещения элемента управления типа `Image`, содержащего изображение автомобиля, движущегося по пути, рассчитанному в прикладной программе глобальной системы местоопределения (GPS).

Траектория движения создается в среде Expression Blend довольно просто. В качестве примера создайте новый проект приложения WPF, присвоив ему имя `WPFMotionPathApp`. Как и при построении специального объекта раскадровки, ваша первая задача в данном случае — определить те объекты, которые должны перемещаться по данной траектории движения. А поскольку для этой цели может быть использован любой объект, создайте несколько разных элементов пользовательского интерфейса по своему усмотрению, например `Ellipse`, `Button` и `Rectangle`. Затем присвойте каждому из этих объектов подходящее имя в поле Name на панели Properties, например `circleOne`, `btnClickMe` (кнопка "Щелкни на мне") и `myRect`.

После определения ряда объектов в качестве целевых для анимации по траектории ваша следующая задача — создать траектории движения. Для этой цели можете выбрать соответствующие элементы управления из категории Shapes в библиотеке ресурсов или же нарисовать контуры траекторий произвольной геометрической формы инструментами Pen и Pencil. Итак, создайте три разные геометрические формы на монтажном столе (их местоположение, размеры и конфигурация особого значения не имеют). Но ради поддержания порядка на монтажном столе присвойте каждой созданной форме будущей траектории движения подходящее имя, например `myStar`, `myPolygon` и `myArc`.

Кроме того, для каждой геометрической формы будущей траектории движения (в данном случае звезды, многоугольника и дуги) рекомендуется настроить свойство `Fill` на режим раскраски No brush в редакторе кистей, а также установить значение в пределах 2-3 в свойстве `StrokeThickness`, доступном в области Appearance на панели Properties, чтобы траектории движения было лучше видно. В качестве примера на рис. 3.15 приведено главное окно (объект типа `Window`) на данном этапе разработки проекта на платформе WPF.

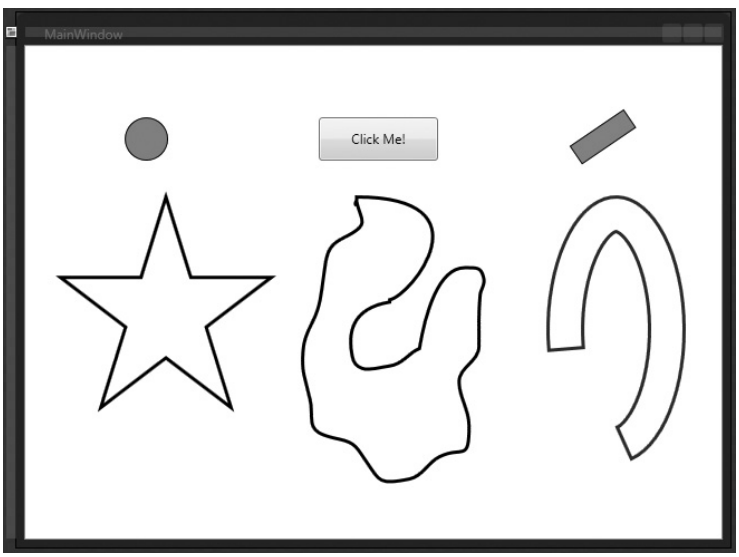

Рис. 3.15. Три элемента пользовательского интерфейса и три контура, определяющие анимацию по траектории

Выберите на панели Objects and Timeline контур первой формы, чтобы преобразовать его в траекторию движения (в данном случае это контур в форме звезды). Щелкните правой кнопкой мыши на узле этого объекта в иерархическом представлении и выберите команду Path⇨Convert to Motion Path (Контур⇨Преобразовать в траекторию движения) из всплывающего контекстного меню, как показано на рис. 3.16.

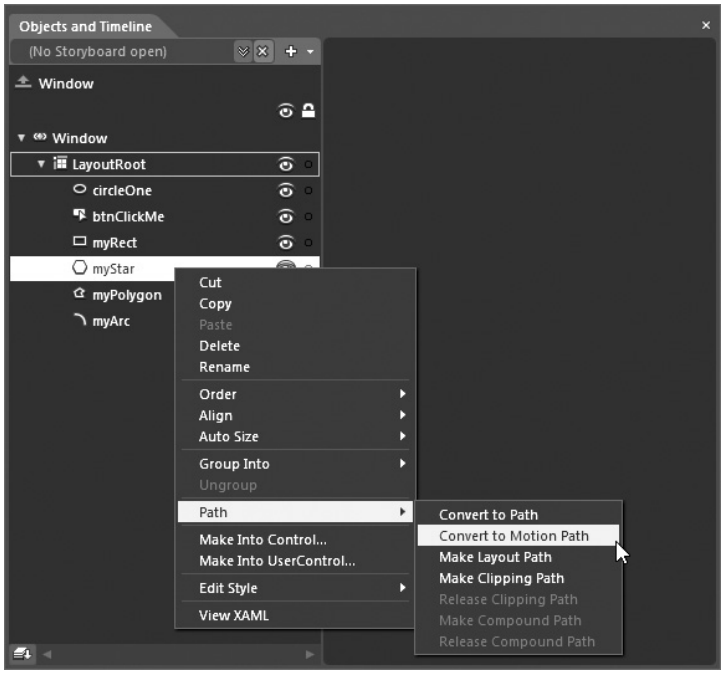

Рис. 3.16. Преобразование контура в траекторию движения

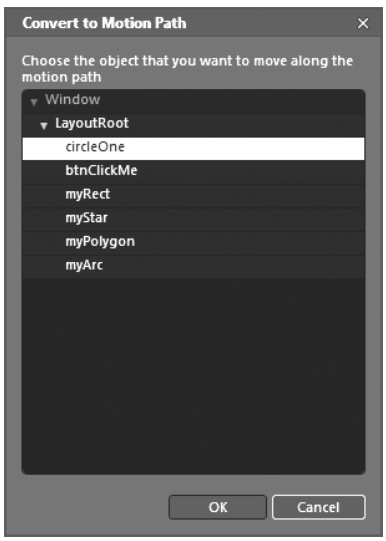

Рис. 3.17. Выбор объекта для перемещения по траектории движения

После выбора этой команды появится диалоговое окно, в котором можете далее выбрать один или несколько объектов для перемещения по траектории движения, преобразуемой из контура заданной формы. В данном случае выберите круг в качестве одного из целевых объектов, как показано на рис. 3.17.

Как только вы выберете целевой объект, будет создан новый объект раскадровки `Storyboard1` (он появится в верхней части иерархического представления на панели Objects and Timeline). Переименуйте этот объект описанным ранее способом в `MoveShapes` (перемещение форм). Обратите внимание на то, что после создания раскадровки вы сможете воспользоваться любым из рассмотренных ранее в этой главе способов для настройки свойств этой раскадровки (режима автоматического изменения на обратное направление воспроизведения анимации, продолжительности анимации и т.д.). Кроме того, можете изменить начальный и конечный моменты каждого промежутка времени между ключевыми кадрами.

Настройте анимационную последовательность таким образом, чтобы она начиналась на нульсекундной отметке и оканчивалась на пятисекундной. Таким образом, вращательное движение круга по траектории в форме звезды будет продолжаться около 5 секунд.

Повторите описанную выше общую процедуру для перемещения второго и третьего целевых объектов (в данном случае это объекты типа `Button` и `Rectangle`) по остальным траекториям движения. Измените по своему усмотрению начальный и конечный моменты анимации перемещения каждого из этих объектов по траектории движения. В качестве примера на рис. 3.18 показано, что общая продолжительность всей анимации составляет восемь секунд, хотя движение каждого целевого объекта (круга, кнопки и прямоугольника) начинается и оканчивается в разные моменты времени.

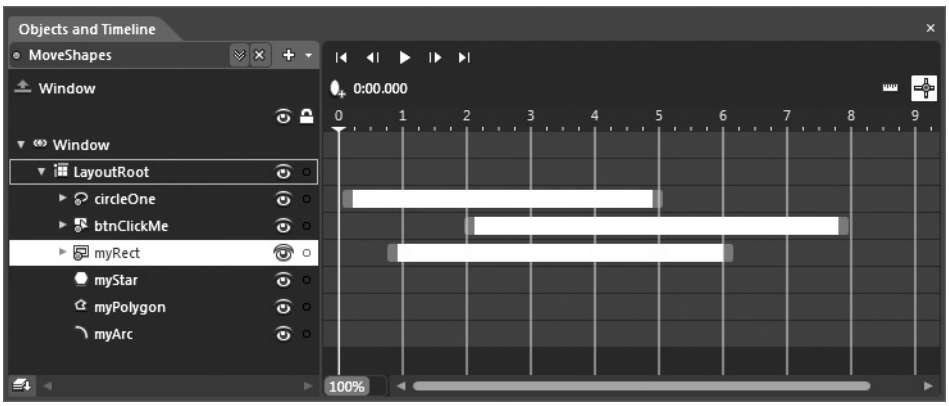

Рис. 3.18. Анимация движения каждого целевого объекта по заданной траектории

Для того чтобы сделать рассматриваемый здесь пример более интересным, организуйте на панели Properties обработку события `Click`, связанного с объектом типа `Button`. Так, если щелкнуть на этом объекте, появится информативное окно сообщения `You clicked me!` (Вы щелкнули на мне!), как показано в приведенном ниже коде обработчика событий.

```
private void btnClickMe_Click(object sender,
  System.Windows.RoutedEventArgs e)
{
   MessageBox.Show("You clicked me!");
}
```

Запустите приложение на выполнение, нажав функциональную клавишу <F5> или комбинацию клавиш <Ctrl+F5>. В итоге все три объекта автоматически последуют по заданным траекториям движения. (Проверьте свою реакцию: успеете ли вы щелкнуть на кнопке, пока она движется.) В отличие от рассмотренного ранее примера проекта на платформе Silverlight, в данном случае анимация начинается, как только главное окно приложения (объект типа `Window`) загрузится в оперативную память. Такое поведение объясняется тем, что по умолчанию анимация по любой раскадровке, создаваемой в области действия приложения WPF, автоматически начинается после загрузки главного окна благодаря приведенной ниже разметке XAML из файла `MainWindow.xaml`.

```
<Window.Triggers>
  <EventTrigger RoutedEvent="FrameworkElement.Loaded">
    <BeginStoryboard x:Name="MoveShapes_BeginStoryboard"
                  Storyboard="{StaticResource MoveShapes}"/>
  </EventTrigger>
</Window.Triggers>
```

Найдите эту разметку, используя редактор XAML, доступный на монтажном столе. Как видите, в этой разметке описывается применяемый на платформе WPF триггер, с помощью которого, как известно, перехватывается условие наступления события, что избавляет от необходимости писать для этой цели специальный код на C# или VB. В данном случае триггер проверяет момент наступления события `Loaded`, связанного с загрузкой объекта типа `Window` в оперативную память[6]. Когда это событие наступает, обрабатывается элемент разметки `<BeginStoryboard>` и начинается воспроизведение анимации по раскадровке, представленной объектом `MoveShapes`.

Несмотря на то что данный триггер анимации на платформе WPF оказывается очень удобным в тех случаях, когда требуется быстро проверить, каким образом анимация будет вести себя во время выполнения разрабатываемого приложения, следует ясно отдавать себе отчет в том, что не всякая анимация должна начинаться после загрузки основного окна приложения. Вполне возможно, что вам потребуется начинать анимацию только в том случае, если пользователь выберет конкретный пункт меню, щелкнет на отдельной кнопке или выполнит иное предусмотренное действие. Разумеется, события можно обрабатывать и вручную, начиная анимацию по раскадровке непосредственно в коде, а среда триггеров на платформе WPF предоставляет лишь альтернативный вариант запуска анимации без написания специального кода. Рассмотрим далее инструменты, доступные в среде Expression Blend IDE для применения триггеров в проектах на платформе WPF.

Исходный код. Исходный код примера проекта `WPFMotionPathApp` находится в папке `Ch 3 Code` загружаемого архива примеров проектов к данной книге.

Управление анимацией с помощью триггеров на платформе WPF

Назначение триггеров на платформе WPF было ранее определено как способ реагирования в разметке на условия наступления событий, а в предыдущем примере проекта было продемонстрировано действие простого триггера на практике. Для более подробного ознакомления со средой триггеров на платформе WPF создайте новый проект приложения WPF, присвоив ему имя `WPFAnimationTriggerApp`.

В данном примере проекта вам предстоит создать новую раскадровку анимации, в ходе которой объект типа `Button` совершает полный оборот. Сначала добавьте новый объект типа `Button` на монтажном столе, присвойте ему имя `btnMyButton` и установите подходящее значение в его свойстве `Content`. Затем создайте новую раскадровку, присвоив ей имя `SpinButtonAnimation` (анимация вращения кнопки); напомним, что для этого достаточно щелкнуть на кнопке со знаком + на панели Objects and Timeline. И наконец, определите односекундный промежуток времени на временной шкале, введя два ключевых кадра анимации выбранного элемента управления начиная с нульсекундной отметки, как показано на рис. 3.19.

А теперь выберите объект типа `Button` на монтажном столе и поверните его на 360°, воспользовавшись мышью. Напомним, что графические преобразования объектов можно выполнять непосредственно на монтажном столе. В данном случае поместите курсор мыши с внешней стороны одного из углов кнопки, нажмите кнопку мыши и, не отпуская ее, переместите курсор в нужном направлении вращения кнопки, как показано на рис. 3.20.

---

[6] Класс `Window` расширяет свой родительский класс `FrameworkElement`.

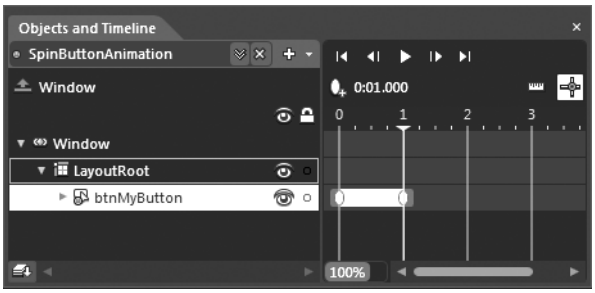

Рис. 3.19. Начальная раскадровка

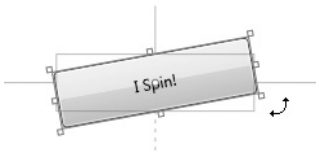

Рис. 3.20. Вращение кнопки, представленной объектом типа `Button`

Если вы теперь запустите приложение на выполнение, элемент управления типа `Button` должен совершить полный оборот при запуске приложения благодаря автоматически формируемому триггеру. Прежде чем исследовать причины такого первоначального поведения, просмотрите всю разметку, автоматически сформированную до сих пор в коде XAML. Эта разметка, немного отредактированная и аннотированная, приведена ниже.

```
<Window ... >
  <!-- Помните! Объекты раскадровки формируются как ресурсы объектов -->
  <Window.Resources>
  <Storyboard x:Key="SpinButtonAnimation">
    ...
  </Storyboard>
  </Window.Resources>

  <!-- Этот триггер запускает анимацию по раскадровке SpinButtonAnimation,
      когда объект типа Window загружается в оперативную память -->
  <Window.Triggers>
    <EventTrigger RoutedEvent="FrameworkElement.Loaded">
      <BeginStoryboard x:Name="SpinButtonAnimation_BeginStoryboard"
        Storyboard="{StaticResource SpinButtonAnimation}"/>
    </EventTrigger>
  </Window.Triggers>

  <Grid x:Name="LayoutRoot">
    <!-- Эта кнопка служит в качестве целевого объекта для анимации -->
    <Button x:Name="btnMyButton" ... >
      ...
    <Button/>
  </Grid>
</Window>
```

Ввод триггера на панели Triggers

Вернитесь сначала к виду рабочего пространства конструирования (Design), нажав функциональную клавишу <F6>, если вы еще не сделали этого, а затем найдите панель Triggers (Триггеры). В рабочем окне Expression Blend IDE она находится рядом с панелью Assets, но ее, как, впрочем, и любую другую панель в этой среде, можно показывать и скрывать по соответствующей команде, выбираемой из меню Window. Как показано на рис. 3.21, на панели Triggers доступен один и только один триггер для текущего объекта типа `Window`, а именно: триггер, запускающий анимацию по раскадровке `SpinButtonAnimation` после загрузки главного окна приложения в оперативную память.

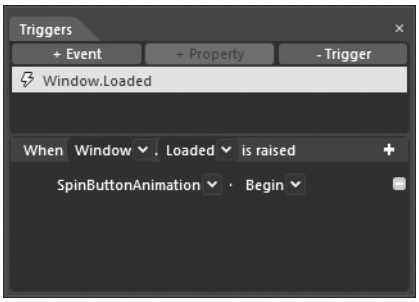

Рис. 3.21. На панели Triggers можно организовать запуск анимации по раскадровке при соблюдении условий наступления определенных событий

В верхней области панели Triggers находятся три выбираемые кнопки, с помощью которых можно взаимодействовать с перечисленными ниже типами триггеров на платформе WPF.

- **Триггер события.** Служит для взаимодействия с временной шкалой анимации при наступлении определенного события, например, `Click` при выполнении щелчка кнопкой мыши на выбранном элементе пользовательского интерфейса. Триггеры событий могут быть установлены для любого объекта, находящегося на монтажном столе.
- **Триггер свойства.** Оказывается полезным при создании специального стиля оформления или шаблона (подробнее об этом — в главе 5). По существу, такой механизм позволяет изменять значение одного свойства в зависимости от того, насколько изменяется значение другого свойства.

С помощью кнопки +Event можно вводить новый триггер события, с помощью кнопки +Property — новый триггер свойства, запускающий анимацию по раскадровке при изменении свойства выбранного объекта, а с помощью кнопки –Trigger — удалять выбранный в настоящий момент триггер из редактора или разметки. В этой главе не рассматривается назначение триггеров свойств, поэтому не будем пока что обращать внимания на кнопку +Property, а более подробно триггеры рассматриваются в главе 5.

Глядя на настройку текущего триггера, приведенную на рис. 3.21, можно заметить, что этот триггер отслеживает событие `Window.Loaded`, обозначенное знаком молнии слева от его имени. Ниже списка текущих триггеров находится ряд раскрывающихся списков, с помощью которых составляется простое и удобочитаемое предложение, описывающее то, что происходит при срабатывании триггера. Ниже приведен общий шаблон такого предложения.

```
When objectName.eventOnObject is raised storyboardName.storyboardAction
```

В настоящий момент это предложение гласит: “При наступлении события **`objectName.eventOnObject`** запускается анимация по раскадровке `storyboardName.storyboardAction`”. Итак, удалите текущий триггер полностью, выделив элемент списка `Window.Loaded` и щелкнув на кнопке –Trigger.

А теперь введите новый триггер, который начнет анимацию по раскадровке `SpinButtonAnimation`, если щелкнуть на самой кнопке. Для этого щелкните сначала на кнопке +Event, а затем выберите объект типа `Button` по его имени из раскрывающего списка When, где в настоящий момент выбран объект типа `Window`. После этого выберите событие `Click` из раскрывающегося списка is raised, как показано на рис. 3.22.

Примечание. Если не можете найти объект для настройки триггера события на панели Triggers, убедитесь в том, что этот объект выбран в соответствующем узле иерархического представления на панели Objects and Timeline panel! По умолчанию на панели Triggers отображаются варианты выбора только для самого верхнего в иерархии объекта типа `Window` и выбранного в настоящий момент элемента пользовательского интерфейса.

Выбрав объект и условие наступления события, щелкните на кнопке со знаком + — самой последней в определении триггера. В итоге появится дополнительная часть пользовательского интерфейса панели Triggers, где вы можете выбрать любую раскадровку из текущего документа. В данном случае выберите сначала раскадровку `SpinButtonAnimation`, а затем команду, позволяющую начать анимационную последовательность. (Обратите внимание на то, что, помимо команды начать анимацию, можно также выбрать другие команды управления анимацией, как показано на рис. 3.23. Эти команды аналогичны тем, что применяются непосредственно в коде для управления анимацией.)

Если после этого вы запустите свое приложение на выполнение, нажав функциональную клавишу <F5> или комбинацию клавиш <Ctrl+F5>, то обнаружите, что кнопка начнет вращаться только после того, как вы щелкнете на ней. А если вы проанализируете разметку, автоматически сформированную в коде XAML, то заметите, что теперь триггер описывается в ней так, как показано ниже.

```
<Window.Triggers>
  <EventTrigger RoutedEvent="ButtonBase.Click"
                SourceName="btnMyButton">
    <BeginStoryboard
       Storyboard="{StaticResource SpinButtonAnimation}"/>
  </EventTrigger>
</Window.Triggers>
```

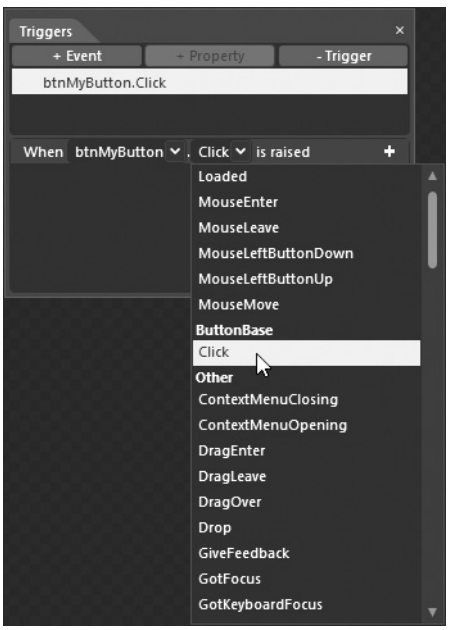

Рис. 3.22. Определение триггера события `Click` для объекта типа `Button`

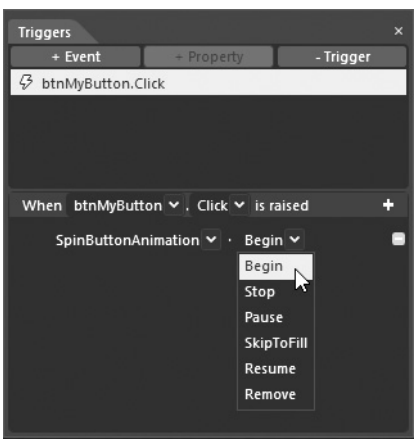

Рис. 3.23. Запуск анимации при наступлении определенного события

Построение системы меню в Expression Blend

В целях дальнейшей демонстрации возможностей триггеров событий нам потребуется расширить рассматриваемый здесь пример проекта, чтобы создать простую систему меню для управления анимационной последовательностью по раскадровке. Прежде всего удалите *полностью* текущий триггер события `Button.Click`. Затем выберите объект раскадровки `SpinButtonAnimation` на панели Objects and Timeline и установите значение `Forever` в свойстве `RepeatBehavior` этого объекта на панели Properties.

Далее воспользуйтесь панелью Assets, чтобы найти элемент управления типа `Menu` и дважды щелкните на нем, чтобы добавить его экземпляр на монтажном столе. Присвойте этому элементу управления имя `mainMenuSystem` (система главного меню) на панели Properties. Измените мышью размеры этого экземпляра объекта типа `Menu` таким образом, чтобы расположить его по всей ширине окна. Щелкните правой кнопкой мыши на данном элементе управления типа `Menu` непосредственно на монтажном столе и выберите команду Add MenuItem (Добавить пункт меню) из всплывающего контекстного меню, как показано на рис. 3.24.

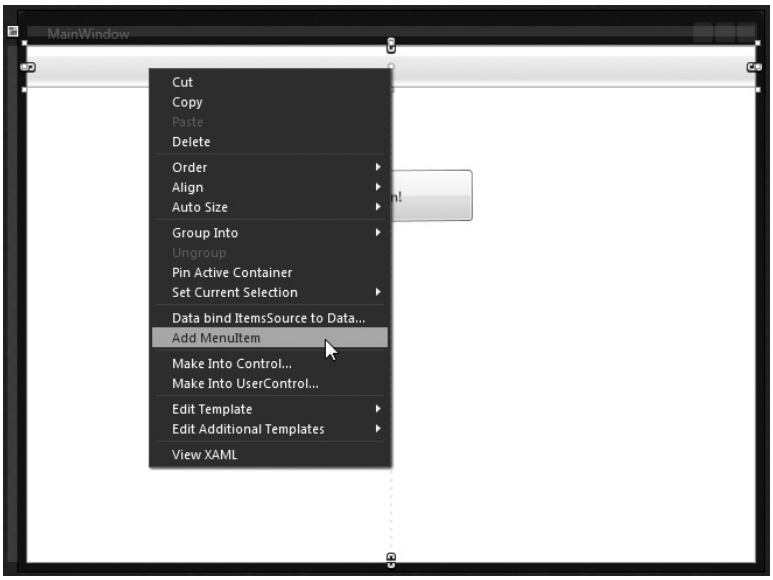

Рис. 3.24. Добавление пункта в меню

Выберите новый элемент управления типа `MenuItem`, найдите его свойство `Header` (Заголовок) на панели Properties и установите в нем значение `File` (свойства на этой панели обнаруживаются быстрее всего в режиме поиска, но в качестве подсказки заметим, что свойство `Header` находится в области Common Properties). Присвойте данному элементу управления имя `mnuFile` (меню файлов).

Примечание. Редактор меню в Expression Blend действует подобно аналогичному редактору в Visual Studio. В частности, щелкнув правой кнопкой мыши на каком угодно объекте типа `MenuItem` непосредственно на монтажном столе, вы сможете добавить к нему пункты подменю и разделительные полосы из всплывающего контекстного меню, как показано на рис. 3.24. А если вы выберете какой угодно объект типа `MenuItem` на панели Objects and Timeline, то сможете затем настроить любые его свойства и организовать обработку любого количества событий, связанных с этим объектом.

Еще раз выберите объект `mnuFile` на монтажном столе и щелкните на нем правой кнопкой мыши, чтобы добавить три пункта подменю. В целях рассматриваемого здесь примера присвойте свойствам `Header` этих элементов меню значения `Play!` (Воспроизвести!), **`Stop!`** (Остановить!) и **`Pause!`** (Приостановить!), а самим элементам — имена `mnuPlay`, `mnuStop` и `mnuPause` соответственно. По завершении построенная вами система меню должна выглядеть так, как показано на рис. 3.25.

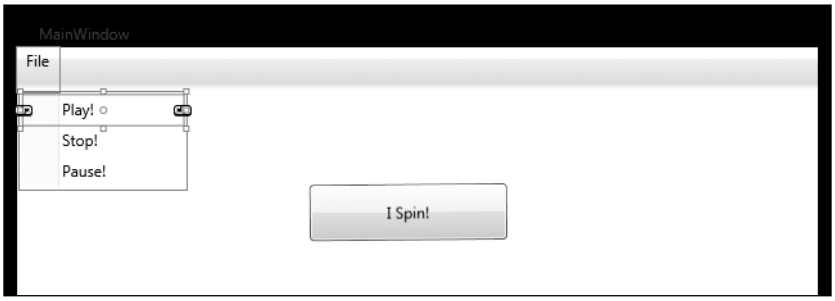

Рис. 3.25. Построение системы меню в Expression Blend

Обратите внимание на вложенный характер каждого компонента меню, отображаемого на панели Objects and Timeline (рис. 3.26). А теперь, когда у нас имеется элементарная система меню, мы можем приступить к созданию некоторых триггеров событий.

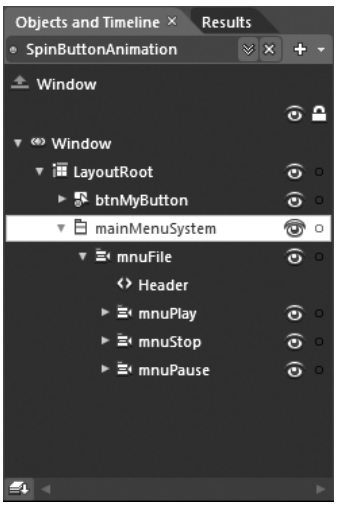

Рис. 3.26. Иерархическое представление системы вложенных меню на панели Objects and Timeline

Присваивание триггеров отдельным пунктам меню

Выберите объект `mnuPlay` на панели Objects and Timeline, перейдите к панели Triggers и щелкните на кнопке +Event. Воспользуйтесь редактором триггеров на панели Triggers, чтобы запускать анимацию по раскадровке `SpinButtonAnimation` при выборе пункта меню File⇨Play!. Аналогичным образом организуйте остановку и приостановку анимации при выборе пунктов меню File⇨Stop! и File⇨Pause! соответственно. На рис. 3.27 показан окончательно установленный ряд триггеров событий.

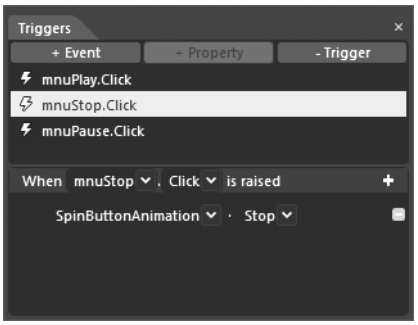

Рис. 3.27. Связывание триггеров событий с соответствующими пунктами меню

Вновь запустите приложение на выполнение и воспользуйтесь меню для управления анимацией по заданной раскадровке. Оцените по достоинству интерактивные возможности своего приложения, созданного без единой строки кода, написанной на C# или VB!

Примечание. Просмотрите ради интереса разметку, автоматически сформированную в коде XAML. Вы обнаружите, что в коллекцию `<Window.Triggers>` теперь входит несколько элементов `<EventTrigger>`, предназначенных для манипулирования различными вариантами управления анимацией по заданной раскадровке.

Для фиксации остальных условий наступления событий, связанных с объектом раскадровки, в меню можно, конечно, добавить дополнительные пункты, а еще лучше — дополнительные элементы пользовательского интерфейса и установить совершенно новые триггеры для других раскадровок. Эти возможности расширить рассматриваемый здесь пример проекта оставляются вам, читатель, в качестве упражнения по освоению на практике триггеров, целевых объектов анимации и ее раскадровки.

Исходный код. Исходный код примера проекта `WPFAnimationTriggerApp` находится в папке `Ch 3 Code` загружаемого архива примеров проектов к данной книге.

Представление об эффектах инерционности движения в анимации

Надеюсь, что, дойдя до этого раздела, вы вполне освоились с основными средствами и способами создания анимации в Expression Blend и теперь ясно представляете себе следующее.

- Общее назначение анимации на платформах WPF и Silverlight, которое состоит в изменении значений свойств объектов во времени.
- Назначение объекта раскадровки, которое состоит в связывании объектов, изменении их свойств и временных промежутков анимации.
- Использование панели Objects and Timeline для создания новой раскадровки и ряда ключевых кадров.
- Использование монтажного стола для изменения значений свойств объектов в режиме редактирования анимации на временной шкале.
- Управление анимацией по раскадровке в коде.
- Управление анимацией по раскадровке с помощью триггеров на платформе WPF.

Безусловно, ясное представление об упомянутых выше средствах и способах создания анимации означает заметный прогресс в освоении среды Expression Blend, но это совсем не означает, что вы можете теперь почивать на лаврах. Ведь имеется ряд дополнительных возможностей для создания анимации в Expression Blend, о которых следует непременно упомянуть. В частности, речь далее пойдет о применении редактора анимации для внедрения в нее *эффектов инерционности движения*. Проще говоря, подобные эффекты позволяют накладывать различные физические ограничения в промежутках времени между двумя последовательными ключевыми кадрами анимации. С помощью эффектов инерционности движения можно вносить в раскадровки такие физические явления, как отскакивание, притягивание, упругость объектов и пр.

Построение исходной компоновки

Как и прежде, начните с создания нового проекта, но на этот раз — приложения Silverlight, присвоив ему имя `AnimationEasingEffects`[7]. Как и в остальных примерах проектов, рассматривавшихся в этой главе, ваша первая задача — определить ряд объектов, а затем — ряд раскадровок для их анимации. Следуя рассматриваемому здесь примеру, нарисуйте три элемента пользовательского интерфейса на монтажном столе и присвойте им подходящие имена на панели Properties. Это могут быть любые графические элементы, но для данного примера выбраны формы круга, треугольника и многоугольника.

Следует иметь в виду, что на этот раз объекты должны располагаться у самого верхнего края монтажного стола, поскольку в результате применения к ним различных эффектов инерционности движения они будут отпущены для свободного падения вниз к нижнему краю экрана. Нарисуйте далее инструментом Pencil зону “приземления” объектов у нижнего края монтажного стола. На рис. 3.28 показан текущий вид компоновки пользовательского интерфейса в рассматриваемом здесь примере проекта приложения Silverlight.

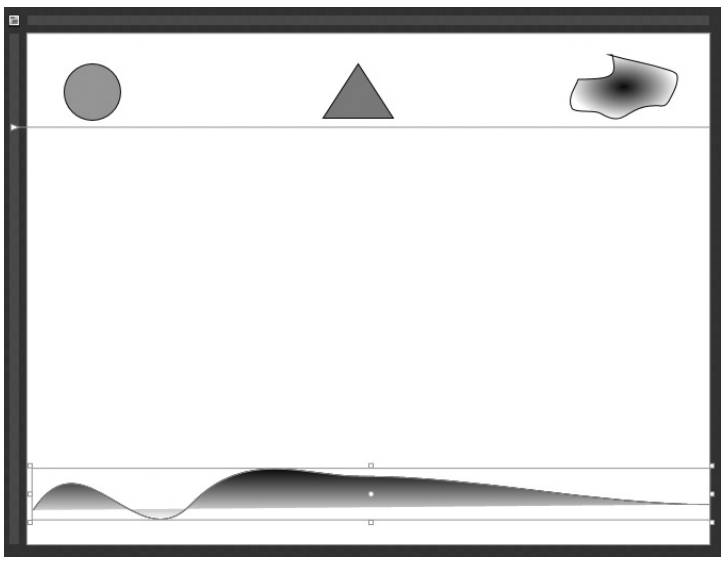

Рис. 3.28. Объекты, готовые к свободному падению

[7] В проектах WPF, разрабатываемых в среде Expression Blend IDE, также поддерживаются инструменты для организации инерционности движения в анимации, поэтому рассматриваемые далее способы в равной степени применимы к настольным приложениям, создаваемым на платформе Windows Presentation Foundation.

Создание исходных раскадровок

Следующая ваша задача состоит в создании отдельных раскадровок `DropAndBounceBall` (падающий и отскакивающий шар), `RubberbandTriangle` (треугольник, растягивающийся, как резиновая лента) и `HoverAndCrashPoly` (повисающий и разбивающийся многоугольник). В этой анимационной последовательности будет воспроизведено движение объекта типа `Ellipse` от исходной точки к зоне приземления. Итак, применяя способы, рассматривавшиеся в этой главе, настройте раскадровку `DropAndBounceBall` для фиксации этого движения в течение четырех секунд. Для этого просто переместите объект типа `Ellipse` в новое положение в режиме регистрации его движения.

Аналогичным образом создайте раскадровки `RubberbandTriangle` и `HoverAndCrashPoly`. Для этого переместите два оставшихся объекта от исходной точки к зоне приземления в течение четырех секунд (опять же в режиме регистрации их движения).

По завершении проверьте полученные в итоге анимационные последовательности, выбрав сначала раскадровку по ее имени на панели Objects and Timeline, а затем щелкнув на кнопке Play. Удовлетворившись первоначальными результатами анимации, перейдите к следующему разделу, где будет показано, каким образом применяются эффекты инерционности движения.

Применение эффектов инерционности движения в анимации

Выберите раскадровку `DropAndBounceBall` на панели Objects and Timeline и щелкните на кнопке с яйцевидной пиктограммой, обозначающей завершающий ключевой кадр анимации на временной шкале. Аналогично выбору объекта на монтажном столе, после щелчка непосредственно на нужном ключевом кадре в редакторе анимации на панели Properties появляется ряд настраиваемых свойств[8]. Как только вы щелкнете на ключевом кадре, на панели Properties появится область Easing (Инерционность движения), где по умолчанию выбирается кнопка EasingFunction (Функция инерционности движения). В настоящий момент ни один из эффектов инерционности движения в данном ключевом кадре не применяется, как показано на рис. 3.29.

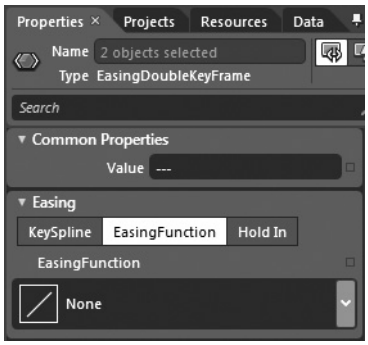

Рис. 3.29. Настройка ключевого кадра на эффекты инерционности движения

Каждый ключевой кадр может быть настроен на самые разные встроенные в Expression Blend эффекты инерционности движения, и все они выбираются из раскры-

[8] Имейте в виду, что отдельным ключевым кадрам можно присваивать имена, используя свойство `Name`. Это дает возможность управлять ключевыми кадрами и их свойствами непосредственно в коде.

вающегося списка EasingFunction. Как показано на рис. 3.30, все эффекты инерционности движения разделены на три категории.

- Варианты, перечисленные в столбце In, означают применение выбранного эффекта в начальной части ключевого кадра.
- Варианты, перечисленные в столбце Out, означают применение выбранного эффекта в завершающей части ключевого кадра.
- Варианты, перечисленные в столбце InOut, означают применение выбранного эффекта как в начальной, так и в завершающей части ключевого кадра.

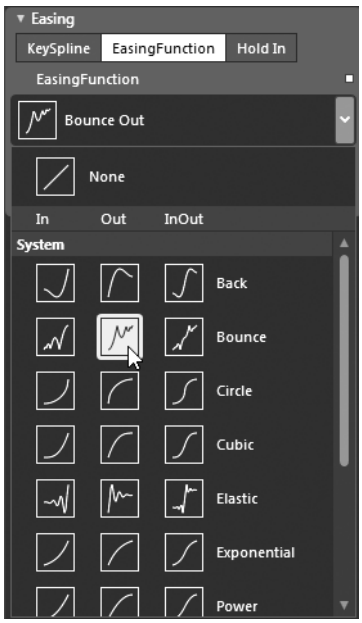

Рис. 3.30. Различные категории эффектов инерционности движения

Если вы раскроете список EasingFunction, то обнаружите в нем целый ряд встроенных в Expression Blend эффектов инерционности движения и их функций. Итак, выберите для данного ключевого кадра раскадровки `DropAndBounceBall` эффект Bounce (Отскакивание) в столбце Out (В конце), как показано на рис. 3.30.

Выбрав нужный эффект инерционности движения, можете далее настроить его на панели Properties. В качестве примера на рис. 3.31 показано, каким образом настраивается количество отскоков (свойство `Bounces`) и их интенсивность (свойство `Bounciness`).

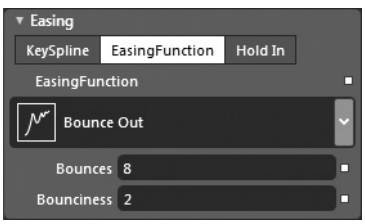

Рис. 3.31. Настройка свойств эффекта инерционности движения `Bounce Out`

Предварительно убедившись в том, что на панели Objects and Timeline выбрана раскадровка `DropAndBounceBall`, проверьте применение эффекта инерционности движения в анимации падающего и отскакивающего шара, щелкнув на кнопке Play. Продолжите экспериментирование с эффектом Bounce Out, а еще лучше — опробуйте в ознакомительных целях ряд других эффектов.

Выберите вторую раскадровку (в данном примере — `RubberbandTriangle`) и щелкните на завершающем ключевом кадре анимации. На этот раз попробуйте применить эффект инерционности движения `Elastic In` (Упругость в начале), у которого имеются интересные свойства `Oscillations` (Колебания) и `Springiness` (Пружинистость), как показано на рис. 3.32.

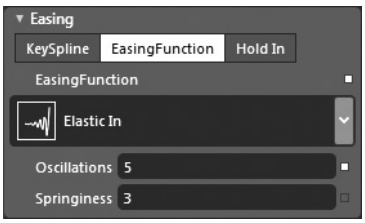

Рис. 3.32. Настройка свойств эффекта инерционности движения `Elastic In`

И на этот раз проверьте полученные результаты, щелкнув на кнопке Play панели Objects and Timeline.

Работа с редактором ключевых сплайнов

Преимущество встроенных в Expression Blend эффектов инерционности движения в анимации заключается в том, что они позволяют оперативно внедрять распространенные физические явления и свойства в анимационную последовательность. Если же вам требуется более точная настройка подобных эффектов, воспользуйтесь для этой цели редактором ключевых сплайнов (KeySpline). В качестве упражнения выберите объект завершенной раскадровки на панели Objects and Timeline и щелкните сначала на последнем ключевом кадре анимации, а затем на кнопке KeySpline. В итоге откроется новая область с редактором ключевых сплайнов, где можно откорректировать изменение скорости движения объекта по мере его приближения к данному ключевому кадру. Итак, откорректируйте значения координат `x1`, `x2`, `y1` и `y2`, переместив отдельные ползунковые элементы управления или щелкнув и перетащив желтую управляющую точку, но принимая во внимание следующее.

- Чем круче линия ключевого сплайна, тем быстрее изменяется значение в данной точке.
- Когда линия ключевого сплайна становится прямой, проходящей из левого нижнего угла графика к правому верхнему его углу, интерполяция получается линейной во времени.

На рис. 3.33 приведен пример коррекции формы ключевого сплайна, в результате которой последний целевой объект анимации в рассматриваемом здесь примере (т.е. многоугольник) сначала повисает на полпути к зоне приземления, а затем разбивается о нее.

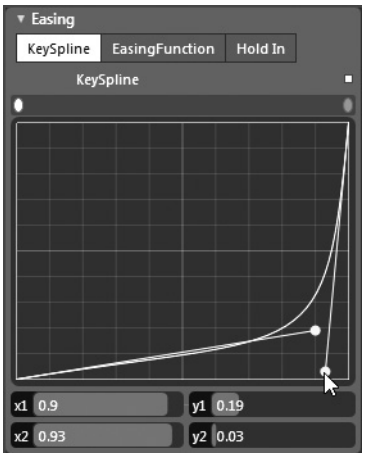

Рис. 3.33. Применение редактора ключевых сплайнов для точной настройки эффекта повисания и разбивания многоугольника

Воспроизведение анимации по раскадровке во время выполнения

Напомним, что при запуске приложения Silverlight, в отличие от приложения WPF, объекты раскадровки не настраиваются автоматически на начало анимации. В завершение рассматриваемого здесь примера проекта организуем обработку события `MouseLeftButtonDown`, наступающего при нажатии левой кнопки мыши для каждого объекта, который подлежит анимации. С этой целью выберите объекты трех геометрических форм по очереди на панели Objects and Timeline, щелкните сначала на кнопке Events, доступной на панели Properties, а затем дважды на текстовом поле справа от метки события `MouseLeftButtonDown`. После создания всех трех обработчиков событий внесите коррективы в их исходный код, как показано ниже на примере обработчика событий типа `MouseLeftButtonDown`, начинающего анимацию падающего и отскакивающего шара после щелчка левой кнопкой мыши на данном объекте.

```
private void ellipse_MouseLeftButtonDown(object sender,
    System.Windows.Input.MouseButtonEventArgs e)
{
  Storyboard sb = (Storyboard)this.Resources["DropAndBounceBall"];
  sb.Begin();
}
```

Если вы запустите теперь свое приложение Silverlight на выполнение, то, для того чтобы начать воспроизведение анимации, достаточно будет щелкнуть на любой из трех геометрических форм. Напомним, что совсем не обязательно ждать завершения одной анимации, чтобы начать другую. Можно воспроизвести одновременно все анимационные последовательности.

Исходный код. Исходный код примера проекта `AnimationEasingEffects` находится в папке `Ch 3 Code` загружаемого архива примеров проектов к данной книге.

Дальнейшее изучение эффектов инерционности движения в анимации

Надеюсь, вы согласитесь с тем, что описывать порядок настройки каждого эффекта инерционности движения в анимации нет никакого смысла. Ведь количество настроек каждого из подобных эффектов довольно велико, а проиллюстрировать конечный результат их применения в печатном издании все равно нельзя. Поэтому дальнейшее изучение различных эффектов инерционности движения в анимации вам придется продолжить самостоятельно, обратившись за справкой к разделу "Изменение интерполяции анимации в промежутках между ключевыми кадрами" (Change animation interpolation between keyframes) руководства пользователя Expression Blend (рис. 3.34).

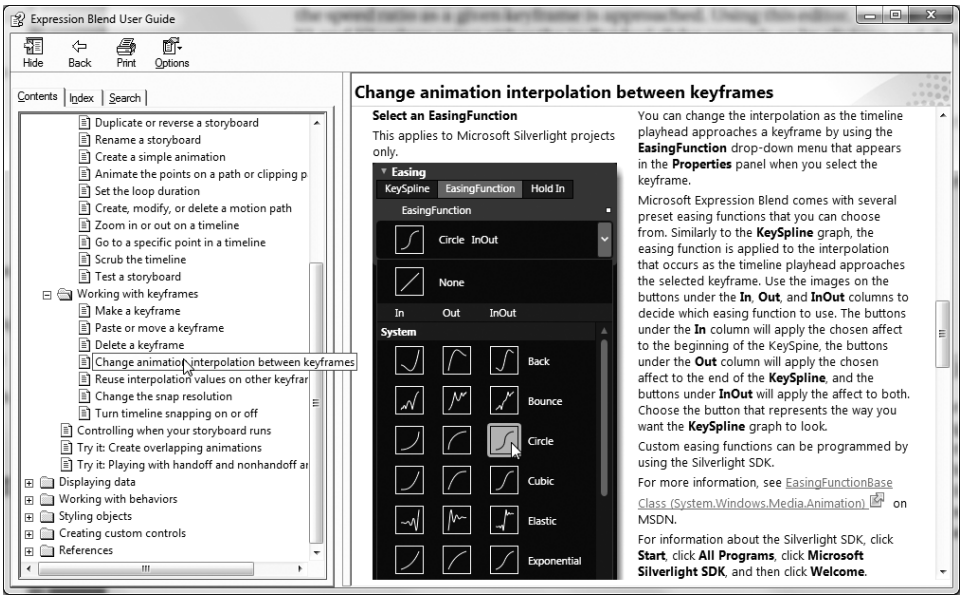

Рис. 3.34. За дополнительными сведениями о различных эффектах инерционности движения в анимации обращайтесь к руководству пользователя Expression Blend

Управление анимацией в разметке XAML с помощью объектов поведения

В завершение этой главы, посвященной средствам и способам создания анимации в Expression Blend, нельзя не упомянуть хотя бы вскользь о назначении *объектов поведения*, более подробно рассматриваемых в главе 4. По существу, объект поведения дает возможность применять сложную логику к элементу пользовательского интерфейса во время выполнения, вообще не прибегая к написанию процедурного кода.

Как будет показано в главе 4, существуют различные объекты поведения, с помощью которых можно перемещать мышью отдельные элементы пользовательского интерфейса, вызывать методы для этих объектов и выполнять прочие действия. Далее речь пойдет об объекте поведения типа `ControlStoryboardAction`, предоставляющем все необходимые средства для того, чтобы начинать, останавливать и приостанавливать анимацию в разметке XAML, причем делать это вне всякой связи со средой триггеров на платформе WPF.

Видоизмененный пример проекта SimpleBlendAnimations

Для демонстрации назначения и функциональных возможностей упомянутого выше объекта поведения еще раз откройте в среде Expression Blend IDE проект SimpleBlendAnimations, рассматривавшийся в качестве первого примера в этой главе, сохраните его полную копию по команде меню File⇨Save Copy of Project (Файл⇨Сохранить копию проекта) и присвойте новой копии имя SimpleBlendAnimations _ Behavior. Убедитесь в том, что новая копия данного проекта открыта теперь в среде Expression Blend IDE.

А теперь откройте в редакторе XAML файл разметки MainPage.xaml и найдите в разметке определение кнопки Start Animation! (Начать анимацию). В этом определении должно присутствовать значение обработчика событий Click, как показано ниже.

```
<Button x:Name="btnStartAnimation" Content="Start Animation!" ...
  Click="btnStartAnimation_Click"/>
```

Удалите значение обработчика событий Click полностью из определения упомянутой выше кнопки, чтобы разметка XAML выглядела в конечном итоге так, как показано ниже (конкретные свойства кнопки могут отличаться в зависимости от настройки данного объекта).

```
<Button x:Name="btnStartAnimation" Content="Start Animation!"
    HorizontalAlignment="Left" Margin="8,8,0,0"
    VerticalAlignment="Top" Width="124"/>
```

Откройте файл исходного кода C# и удалите полностью логику обработчика событий Click, где содержится код программного запуска анимации по раскадровке. Иными словами, удалите из исходного кода все следы метода btnStartAnimation _ Click(). Для проверки на отсутствие ошибок компиляции постройте и запустите рассматриваемый здесь проект на выполнение.

Добавление в проект объекта поведения типа ControlStoryboardAction

Перейдите к визуальному конструктору основной страницы, описываемой в разметке из файла MainPage.xaml, на монтажном столе, откройте панель Assets, выберите категорию Behaviors (Виды поведения) и найдите объект типа ControlStoryboardAction (рис. 3.35).

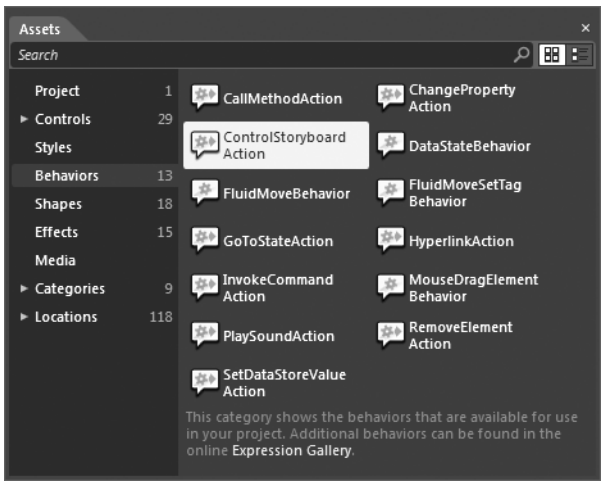

Рис. 3.35. Объект поведения типа ControlStoryboardAction

Перетащите этот объект поведения на объект типа `Button`, находящийся на монтажном столе (рис. 3.36). С другой стороны, можете перетащить объект поведения на соответствующий узел иерархического представления на панели Objects and Timeline.

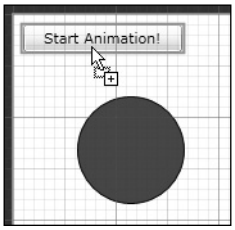

Рис. 3.36. Отдельные виды поведения можно перетаскивать в виде объектов на монтажный стол

Выберите вновь созданный узел на панели Objects and Timeline (в иерархическом представлении он должен появиться ниже узла `Button`), чтобы исследовать свойства, доступные в области Trigger на панели Properties, как показано на рис. 3.37. Обратите внимание на то, что в свойстве `SourceObject` (Исходный объект) устанавливается имя объекта типа `Button` вследствие предыдущей операции перетаскивания. В этом свойстве указывается тот объект на монтажном столе, с которым должен взаимодействовать объект поведения. Обратите также внимание на то, что в свойстве `EventName` (Имя события) автоматически устанавливается событие типа `Click`, хотя из раскрывающегося списка можно выбрать любое другое событие.

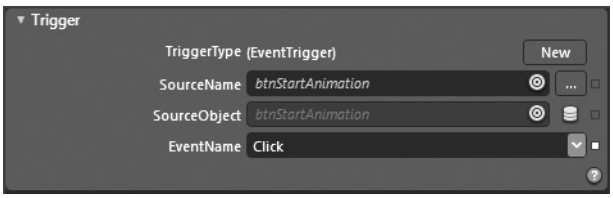

Рис. 3.37. Свойства `SourceName` и `EventName` служат для настройки порядка инициирования объектом конкретного поведения

Для того чтобы завершить настройку данного конкретного поведения, остается лишь выбрать сначала манипулируемый объект раскадровки, а затем указать действие, которое должно выполняться при срабатывании триггера соответствующего события. В качестве примера на рис. 3.38 показано, что если щелкнуть на кнопке, находящейся на главной странице приложения, начнется воспроизведение анимации круга по раскадровке, определяемой объектом `AnimateCircle`.

Запустите свое приложение на выполнение. Теперь у вас имеется возможность начинать анимацию без всякой помощи специального кода C# или обработчиков событий! Если вы проанализируете разметку, автоматически сформированную в коде XAML, то обнаружите в ней приведенный ниже фрагмент.

```
<Button x:Name="btnStartAnimation" Content="Start Animation!"
      HorizontalAlignment="Left" Margin="8,8,0,0"
      VerticalAlignment="Top" Width="124">
 <i:Interaction.Triggers>
   <i:EventTrigger EventName="Click">
   <ei:ControlStoryboardAction
```

```
      Storyboard="{StaticResource AnimateCircle}"/>
    </i:EventTrigger>
  </i:Interaction.Triggers>
</Button>
```

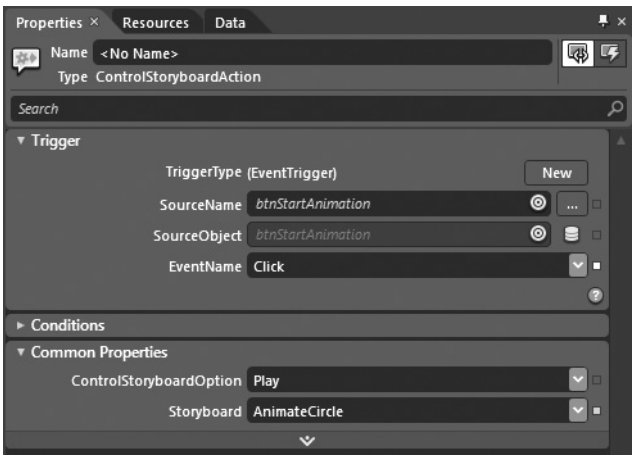

Рис. 3.38. Полностью настроенный объект поведения типа `ControlStoryboardAction`

По желанию можете добавить на монтажном столе ряд дополнительных объектов типа `Button` и определить для них новые объекты поведения типа `ControlStoryboardAction`, чтобы останавливать, приостанавливать и возобновлять циклическое воспроизведение анимации.

Исходный код. Исходный код примера проекта `impleBlendAnimations _ Behavior` находится в папке `Ch 3 Code` загружаемого архива примеров проектов к данной книге.

Резюме

В этой главе было показано, как пользоваться встроенным в Expression Blend редактором анимации, в котором можно создавать объекты раскадровки для приложений на платформах WPF и Silverlight. Напомним, что раскадровка состоит из ряда ключевых кадров анимации, в которых значения свойств объектов изменяются во времени. После рассмотрения различных элементов рабочего пространства анимации в этой главе было показано, каким образом организуется взаимодействие с раскадровкой в коде. В частности, получив нужный ресурс, можно далее воспользоваться средствами, предоставляемыми в классе `Storyboard`, для управления анимацией.

Далее в главе были рассмотрены некоторые средства и способы создания анимации на платформе WPF, начиная с назначения траекторий движения, по которым перемещаются целевые объекты, а затем перейдя к общему представлению о среде триггеров на платформе WPF. Как было показано на конкретных примерах, триггеры предоставляют возможность взаимодействовать с изменяющимися свойствами и наступающими событиями непосредственно в разметке для управления анимацией по раскадровке, а более подробно о триггерах речь пойдет в последующих главах.

И в заключение главы были вкратце представлены эффекты инерционности движения в анимации и объекты поведения. С помощью встроенных в Expression Blend эффектов инерционности движения можно очень просто (и довольно быстро) применять

физические явления и свойства в создаваемой анимации, в том числе отскакивание, притягивание, упругость и т.д. Для более точного управления подобными эффектами имеется также возможность настроить временные характеристики анимации по отдельным ключевым кадрам в интегрированном в Expression Blend редакторе ключевых сплайнов. А с помощью объектов поведения типа `ControlStoryboardAction` можно полностью манипулировать анимацией непосредственно в разметке XAML и вне всякой связи со средой триггеров на платформе WPF.